

AD-A019 641

PRODUCTION RULES AS A REPRESENTATION FOR A KNOWLEDGE-
BASED CONSULTATION PROGRAM

Randall Davis, et al

Stanford University

Prepared for:

Advanced Research Projects Agency

October 1975

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

027065

Stanford Artificial Intelligence Laboratory
Memo AIM-266

October 1975

Computer Science Department
Report No. STAN-CS-75-519

ADA019641

Production Rules as a Representation for a Knowledge-Based Consultation Program

by

Randall Davis
Bruce Buchanan
Edward Shortliffe

Research sponsored by

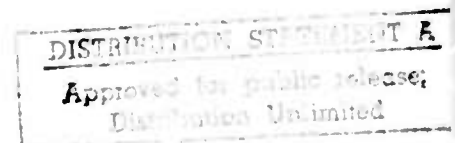
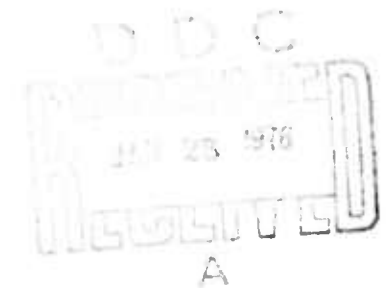
National Institutes of Health
Grant HS01544

Biotechnology Resources
Grant RR-00785

Advanced Research Projects Agency
ARPA Order No. 2494

COMPUTER SCIENCE DEPARTMENT
Stanford University

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER stan-cs-75-519	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Production Rules as a Representation for a Knowledge-Based Consultation Program.		5. TYPE OF REPORT & PERIOD COVERED Technical October 1975
7. AUTHOR(s) R. Davis, B. Buchanan and E. Shortliffe		6. PERFORMING ORG. REPORT NUMBER AIM266
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory Stanford University Stanford, California 94305		8. CONTRACT OR GRANT NUMBER(s) DAHC15-73-C-0435
11. CONTROLLING OFFICE NAME AND ADDRESS Col. Dave Russell, Dep. Dir., ARPA, IPT, ARPA Headquarters, 1400 Wilson Blvd. Arlington, Virginia 22209		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order 2494
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ONR Representative, Philip Surra Durand Aeronautics Building Room 165 Stanford University Stanford, California 94305		12. REPORT DATE October 1975
		13. NUMBER OF PAGES 37 42
		15. SECURITY CLASS. (of this report) 15
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Releasable without limitations on dissemination.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The MYCIN system has begun to exhibit a high level of performance as a consultant on the difficult task of selecting antibiotic therapy for bacteremia. This report discusses issues of representation and design for the system. We describe the basic task and document the constraints involved in the use of a program as a consultant. The control structure and knowledge representation of the system are examined in this light, and special attention is given to the impact of production rules as a representation. The extent of the domain independence of the methodology is also		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

examined.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

October 1975

Computer Science Department
Report No. STAN-CS-75-519

Production Rules as a Representation for a Knowledge-Based Consultation Program

by

Randall Davis
Bruce Buchanan
Edward Shortliffe

ABSTRACT

The MYCIN system has begun to exhibit a high level of performance as a consultant on the difficult task of selecting antibiotic therapy for bacteremia. This report discusses issues of representation and design for the system. We describe the basic task and document the constraints involved in the use of a program as a consultant. The control structure and knowledge representation of the system are examined in this light, and special attention is given to the impact of production rules as a representation. The extent of the domain independence of the methodology is also examined. This is a pre-print of a paper submitted to Artificial Intelligence.

Randall Davis and Bruce Buchanan are in the Computer Science Department, Edward Shortliffe is in the Department of Medicine. The work reported here was funded in part by grants from the National Institutes of Health: NIH Grant HSO1544 and NIH Grant GM 29662, in part by the Advanced Research Projects Agency, under ARPA Contract DAHC15-73-C-0435, and by the Medical Scientist Training Program under NIH Grant GM 10922. Computing support was furnished by the SUMEX-AIM computer resource at Stanford University, funded by the Biotechnology Resources Branch of the National Institutes of Health under Grant RR-00785.

The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, ARPA, National Institutes of Health, or the U. S. Government.

Reproduced in the U.S.A. Available from the National Technical Information Service, Springfield, Virginia 22151.

[1] INTRODUCTION

Two recent trends in artificial intelligence research have been applications of AI to 'real-world' problems, and the incorporation in programs of large amounts of task-specific knowledge. The former is motivated in part by the belief that artificial problems may prove in the long run to be more a diversion than a base to build on, and in part by the belief that the field has developed sufficiently to provide techniques capable of tackling real problems.

The move toward what have been called 'knowledge-based' systems represents a change from previous attempts at generalized problem solvers (as, for example, GPS). Earlier work on such systems demonstrated that while there was a large body of useful general purpose techniques (e.g., problem decomposition into subgoals, heuristic search in its many forms), these did not by themselves offer sufficient power for high performance. Rather than non-specific problem solving power, knowledge-based systems have emphasized both the accumulation of large amounts of knowledge in a single domain, and the development of domain-specific techniques, in order to develop a high level of expertise.

There are numerous examples of systems embodying both trends, including efforts at symbolic manipulation of algebraic expressions [MACSYMA 1974], speech understanding [Lesser 1974], chemical inference [Buchanan 1971], applications of advanced automation techniques to industrial assembly [Finkel 1974, Rosen 1975, Nilsson 1975], some work on natural language [Woods 1972], and the creation of computer consultants as interactive advisors for various tasks [Hart 1975, Shortliffe 1975a].

In this paper we discuss issues of representation and design for one such knowledge-based application program — the MYCIN system developed over the past three years as an interdisciplinary project at Stanford University.¹ We examine in particular how the implementation of various system capabilities is facilitated or inhibited by the use of *production rules* as a knowledge representation. In addition, the limits of applicability of this methodology are investigated.

We begin with a review of features which were seen to be essential to any knowledge-based consultation system, and suggest how these imply specific program design criteria. We note also the additional challenges offered by the use of such a system in a medical domain. This is followed by an explanation of the system structure, and its fundamental assumptions. The bulk of the paper is then devoted to a report of our experience with the benefits and drawbacks of production rules as a knowledge representation for a high performance AI program.



[2] SYSTEM GOALS

The MYCIN system was developed originally to provide consultative advice on diagnosis of and therapy for infectious diseases — in particular, bacterial infections in the blood.² From the start, the project has been shaped by several important constraints. The decision to construct a high performance AI program in the consultant model brought with it several demands. First, the program had to be *useful* if we expected to attract the interest and assistance of experts in the field. The task area was thus chosen partly because of a demonstrated need: for example, in a recent year one of every four people in the U.S. was given penicillin, and almost 90% of those prescriptions were unnecessary [Kagan1973]. Problems such as these indicate the need for more (or more accessible) consultants to physicians selecting antimicrobial drugs. Usefulness also implies competence, consistently high performance, and ease of use. If advice is not reliable, or is difficult to obtain, the utility of the program is severely impaired.

A second constraint was the need to design the program to accommodate a *large and changing body of technical knowledge*. It has become clear that large amounts of task-specific knowledge are required for high performance, and that this knowledge base is subject to significant changes over time [Buchanan1971, Finkel1974, Green1974]. Our choice of a production rule representation was significantly influenced by such features of the knowledge base.

A third demand was for a system capable of handling an *interactive dialog*, and one which was not a "black box." This meant that it had to be capable of supplying coherent explanations of its results, rather than simply printing a collection of orders to the user. This was perhaps the major motivation for the selection of a symbolic reasoning paradigm, rather than one which, for example, relied totally on statistics. It meant also that the "flow" of dialog — the order of questions — should exhibit the sense of purposefulness of the domain, and not be determined by programming considerations. Interactive dialog required, in addition, extensive human engineering features designed to make interaction simple for someone unaccustomed to computers.

The choice of a medical domain brought with it additional demands [Shortliffe1974b]. Speed, access, and ease of use gained additional emphasis, since a physician's time is typically limited. The program also had to fill a need well-recognized by the clinicians who would actually use the system, since the lure of pure technology is usually insufficient. Finally, the program had to be designed with an emphasis on its supportive role as a tool for the physician, rather than as a replacement for his own reasoning process.

Any implementation selected had to meet all these demands. Predictably, some have been met more successfully than others, but all have been important factors in influencing the system's final design.

[3] SYSTEM OVERVIEW

The system is written in INTERLISP, runs on a DEC KI-10 with approximately 130K of nonshared code, and is fast enough to permit convenient real-time interaction. The consultation system (with the required data structures) requires approximately 50K, the explanation system 10K, the question answering system 15K, the rule acquisition system 20K, and the rules themselves approximately 8K (the remainder includes a dictionary, information on drug properties, and various system utility features).

[3.1] THE TASK

The fundamental task is the selection of therapy for a patient with a bacterial infection. Consultative advice is often required in the hospital because the attending physician may not be an expert in infectious diseases, as for example, when a cardiology patient develops an infection after heart surgery. Time considerations compound the problem. A specimen (of blood, urine, etc.) drawn from a patient may show some evidence of bacterial growth within 12 hours, but 24 to 48 hours (or more) are required for positive identification. The physician must therefore often decide, in absence of complete information, whether or not to start treatment and what drugs to use if treatment is required. Both of these may be difficult questions.

The task will become clearer by reviewing the initial and final parts of a sample dialog with the MYCIN system, shown in Figures 1 and 2 (italicized comments at the right are for explanatory purposes, and are not part of the actual dialog). A great deal of attention has been paid to making the interaction easy and natural. Questions have been carefully worded, for example, so as to set up expectations of responses without making them as uninteresting as multiple choice questions. Currently the dialog is directed primarily by the system — patient information can be entered only by answering each question in turn.³

There are several options available to make the interaction more informative. The user can, for instance, ask for a rephrasing of the question or a display of some (or all) of the acceptable responses. If a requested item of information is unavailable, he can respond with UNKNOWN (U, or UNK for short). The user can also modify any answer which he is unsure of by attaching a number indicating his degree of certainty (as, for example, in questions 13 and 15).

The main purpose of the system is to provide advice, and it does so via the *consultation system* (Figure 3). In addition to this performance program, there are three adjunct programs that increase system utility and flexibility: (a) a *question answering* system which handles simple natural language inquiries about either the specific consultation, or about the general knowledge base of the program; (b) an *explanation program* for giving reasons why the performance program has requested an item of information or has drawn a particular conclusion; and (c) a *knowledge acquisition* system for extending the knowledge base through a dialog with an expert.

As described below, each of these four interrelated programs meets one or more of the goals outlined above.

-----PATIENT-1-----

1) Patient's name: (first-last)

****FRED SMITH**

{background data}

2) Sex:

****MALE**

3) Age:

****55**

4) Have you been able to obtain positive cultures from a site at which Fred Smith has an infection?

****YES**

-----INFECTION-1-----

5) What is the infection?

****PRIMARY-BACTEREMIA**

6) Please give the date and approximate time when signs or symptoms of the primary-bacteremia (INFECTION-1) first appeared. (mo/da/yr)

**** MAY 5, 1975**

The most recent positive culture associated with the primary-bacteremia (INFECTION-1) will be referred to as:

-----CULTURE-1-----

7) From what site was the specimen for CULTURE-1 taken?

****BLOOD**

8) Please give the date and time when this blood culture (CULTURE-1) was obtained. (mo/da/yr)

**** MAY 9, 1975**

The first significant organism from this blood culture (CULTURE-1) will be referred to as:

-----ORGANISM-1-----

9) Enter the identity of ORGANISM-1:

****UNKNOWN**

{typically the identity is not known and must be inferred}

10) Is ORGANISM-1 a rod or coccus (etc.):

****ROD**

11) The gram stain of ORGANISM-1:

****GRAMNEG**

12) Have there been POSITIVE cultures from sites which are NOT associated with infections about which you are seeking advice?

****NO**

13) Did ORGANISM-1 grow aerobically?

****YES (8)**

14) Did ORGANISM-1 grow anaerobically?

****YES**

15) What is the suspected portal of entry of ORGANISM-1 into this sterile site?

****GI (6)**

...

{more questions follow, to gather sufficient information to infer possible identities of the infecting organisms}

FIGURE 1

My therapy recommendation will be based on the following probable infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY-BACTEREMIA

{possible identities}

- <Item 1> The identity of ORGANISM-1 may be PSEUDOMONAS-AERUGINOSA
- <Item 2> The identity of ORGANISM-1 may be KLEBSIELLA-PNEUMONIAE
- <Item 3> The identity of ORGANISM-1 may be E.COLI
- <Item 4> The identity of ORGANISM-1 may be BACTEROIDES-FRAGILIS
- <Item 5> The identity of ORGANISM-1 may be ENTEROBACTER
- <Item 6> The identity of ORGANISM-1 may be PROTEUS-NON-MIRABILIS

46) On a scale of 0 to 4, where higher numbers indicate increasing severity how would you rate Fred Smith's degree of sickness?

****3**

47) Does Fred Smith have a clinically significant allergic reaction to any antimicrobial agent?

****NO**

48) Fred Smith's weight in kilograms (or <number> POUNDS):

****70**

{note that 2 drugs cover for 6 potential organisms}

[Rec 1] My preferred therapy recommendation is as follows:

In order to cover for Items <1 2 3 5 6>:

Give: GENTAMICIN

Dose: 119 mg (1.7 mg/kg) q8h IV [or IM] for 10 days

Comments: Modify dose in renal failure

In order to cover for Item <4>:

Give: CLINDAMYCIN

Dose: 595 mg (8.5 mg/kg) q6h IV [or IM] for 14 days

Comments: If diarrhea or other GI symptoms develop, patient should be evaluated for possible pseudomembranous colitis.

Do you wish to see the next choice therapy?

{alternative choices are available}

**** Y**

Which, if any, of the above drugs do you wish eliminated from consideration?

****CLINDAMYCIN**

[Rec 2] Next best therapy recommendation:

In order to cover for Items <2 3 4 5 6>:

Give: CHLORAMPHENICOL

Dose: 1 gm (15 mg/kg) q6h IV for 14 days

Comments: Monitor patient's white count

In order to cover for Item <1>:

Give: GENTAMICIN

Dose: 119 mg (1.7 mg/kg) q8h IV [or IM] for 10 days

Comments: Modify dose in renal failure

FIGURE 2

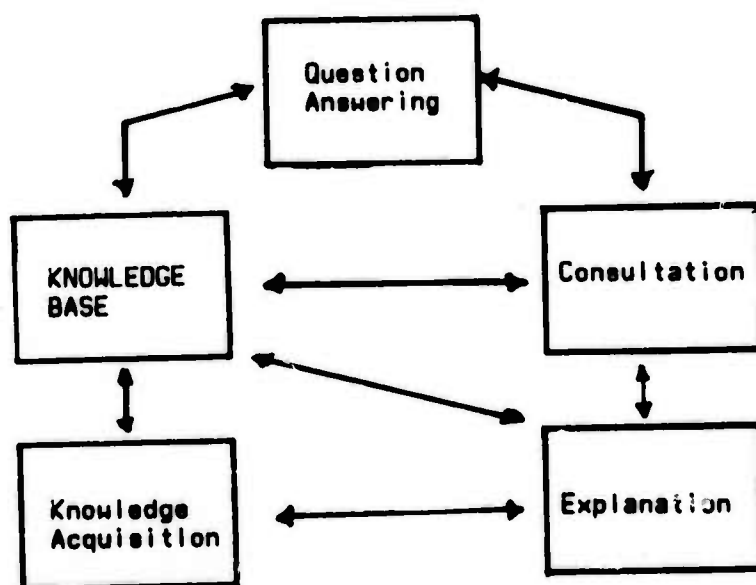


FIGURE 3

[3.2] THE RULES

The primary source of domain specific knowledge is a set of some 200 production rules, each with a premise and an action (Figure 4).

```

PREMISE      ($AND(SAME CNTXT INFECT PRIMARY-BACTEREMIA)
              (MEMBF CNTXT SITE STERILESITES)
              (SAME CNTXT PORTAL GI))

ACTION      (CONCLUDE CNTXT IDENT BACTEROIDES TALLY .7)

```

If

- 1) the infection is primary-bacteremia, and
- 2) the site of the culture is one of the sterilesites, and
- 3) the suspected portal of entry of the organism is the gastro-intestinal tract.

Then there is suggestive evidence (.7) that the identity of the organism is bacteroides.

Figure 4

The PREMISE is a Boolean combination of predicate functions on associative triples. Thus each clause of a premise has the following four components:

<predicate function> <object> <attribute> <value>

There is a standardized set of 24 predicate functions (e.g. SAME, KNOWN, DEFINITE), some 80 attributes (e.g. IDENTITY, SITE, SENSITIVITY), and 11 objects (e.g. ORGANISM, CULTURE, DRUG), currently available for use as primitives in constructing rules. The premise is a conjunction at the top level (top level disjunctions are put into separate rules), but may contain arbitrarily complex conjunctions or disjunctions at lower levels. (We have not found it necessary in practice to use more than two levels.) The ACTION part indicates one or more conclusions which can be drawn if the premises are satisfied, hence the rules are (currently) purely inferential in character.

It is intended that each rule embody a single, modular chunk of knowledge, and state explicitly in the premise all necessary context. Since the rule uses a vocabulary of concepts common to the domain, it forms, by itself, a comprehensible statement of some piece of domain knowledge. As will become clear, this characteristic is useful in many ways.

Each rule is, as is evident, highly stylized, with the IF/THEN format and the specified set of available primitives. While the LISP form of each is executable code (and, in fact, the premise is simply EVALuated by LISP to test its truth, and the action EVALuated to make its conclusions), this tightly structured form makes possible the examination of the rules by other parts of the system. This in turn leads to some important capabilities, to be described below. For example, the internal form can be automatically translated into readable English, as shown in Figure 4.

Despite this strong stylization, we have not found the format restrictive, as evidenced by the fact that of nearly 200 rules on a variety of topics, only 8 employ any significant variations of methodology. The limitations we have encountered arise primarily out of the fact that the use of a simple predicate in the premise forces a pure problem reduction approach (see section [6.1.2]). We

have thus found it necessary, for instance, to use a slightly different technique to implement the few rules which say things like "For each organism such that ... conclude that ...", but this has been done in a way which is closely related to the standard format.

[3.3] JUDGMENTAL KNOWLEDGE

Since we want to deal with real-world domains in which reasoning is often judgmental and inexact, we require some mechanism for being able to say that "A suggests B", or "C and D tend to rule out E." The numbers used to indicate the strength of a rule (e.g. the .7 in Figure 4) have been termed Certainty Factors (CFs). The methods for combining CFs are embodied in a model of approximate implication. Note that while these are derived from and are related to probabilities, they are distinctly different (for a detailed review of the concept, see [Shortliffe1975b]). For the rule in Figure 4, then, the evidence is strongly indicative (.7 out of 1), but not absolutely certain. Evidence confirming an hypothesis is collected separately from that which disconfirms it, and the truth of the hypothesis at any time is the algebraic sum of the current evidence for and against it. This is an important aspect of the truth model, since it makes plausible the simultaneous existence of evidence in favor and against the same hypothesis. We believe this is an important characteristic of any model of inexact reasoning.

Facts about the world are represented as 4-tuples, with an associative triple and its current CF (Figure 5). Positive CFs indicate a predominance of evidence confirming an hypothesis, negative CFs indicate predominance of disconfirming evidence

```
(SITE CULTURE-1 BLOOD 1.0)
(IDENT ORGANISM-2 KLEBSIELLA .25)
(IDENT ORGANISM-2 E.COLI .73)
(SENSITIV ORGANISM-1 PENICILLIN -1.0)
```

Figure 5

Note that the truth model permits the coexistence of several plausible values for a single clinical parameter, if they are suggested by the evidence. Thus, for example, after attempting to deduce the identity of an organism, the system may have concluded (correctly) that there is evidence that the identity is E.coli and evidence that it is Klebsiella, despite the fact that they are mutually exclusive possibilities.

As a result of the program's medical origins, we also refer to the attribute part of the triple as a 'clinical parameter', and use the two terms interchangeably here. The object part (e.g., CULTURE-1, ORGANISM-2) is referred to as a context. This term was chosen to emphasize their dual role as both part of the associative triple and as a mechanism for establishing scope of variable bindings. As explained below, the contexts are organized during a consultation into a tree structure whose function is similar to those found in 'alternate world' mechanisms of languages like QA4.

[3.4] CONTROL STRUCTURE

The rules are invoked in a backward unwinding scheme that produces a depth-first search of an AND/OR goal tree (and hence is similar in some respects to *PLANNER*'s consequent theorems): given a goal to establish, we retrieve the (pre-computed) list of all rules whose conclusions bear on the goal. The premise of each is evaluated, with each predicate function returning a number between -1 and 1. \$AND (the multivalued analogue of the Boolean AND) performs a minimization operation, and \$OR (similar) does a maximization.⁴ For rules whose premise evaluates successfully (i.e. greater than .2, an empirical threshold), the action part is evaluated, and the conclusion made with a certainty which is

$$\langle \text{premise-value} \rangle * \langle \text{certainty factor} \rangle$$

Those which evaluate unsuccessfully are bypassed, while a clause whose truth cannot be determined from current information causes a new subgoal to be set up, and the process recurses. Note that 'evaluating' here means simply invoking the LISP EVAL function — there is no additional rule interpreter necessary, since \$AND, \$OR, and the predicate functions are all implemented as LISP functions.

Unlike *PLANNER*, however, the subgoal which is set up is a generalized form of the original goal. If, for example, the unknown clause is 'the identity of the organism is *E.coli*', the subgoal which is set up is 'determine the identity of the organism.' The new subgoal is therefore always of the form 'determine the value of the <attribute>' rather than 'determine whether the <attribute> is equal to <value>'. By setting up the generalized goal of collecting all evidence about a clinical parameter, the program effectively exhausts each subject as it is encountered, and thus tends to group together all questions about a given topic. This results in a system which displays a much more focussed, methodical approach to the task, which is a distinct advantage where human engineering considerations are important. The cost is the effort of deducing or collecting information which is not strictly necessary. However, since this occurs rarely — only when the <attribute> can be deduced with certainty to be the <value> named in the original goal — we have not found this to be a problem in practice.

A second deviation from the standard rule unwinding approach is that every rule relevant to a goal is used. The premise of each rule is evaluated, and if successful, its conclusion is invoked. This continues until all relevant rules have all been used, or one of them has given the result with certainty. This use of all rules is in part an aspect of the model of judgmental reasoning and the approximate implication character of rules — unless a result is obtained with certainty, we should be careful to collect all positive and negative evidence. It is also appropriate to the system's current domain of application, clinical medicine, where a conservative strategy of considering all possibilities and weighing all the evidence is preferred.

If, after trying all relevant rules (referred to as 'tracing' the subgoal), the total weight of the evidence about a hypothesis falls between -2 and .2 (again, empirically determined), the answer is regarded as still unknown. This may happen if no rule were applicable, the applicable rules were too weak, the effects of several rules offset each other, or if there were no rules for this subgoal at all. In

any of these cases, when the system is unable to deduce the answer, it asks the user for the value (using a phrase which is stored along with the attribute itself). Since the legal values for each attribute are also stored with it, the validity (or spelling) of the user's response is easily checked. (This also makes possible a display of acceptable answers in response to a '?' answer from the user)

The strategy of always attempting to deduce the value of a subgoal, and asking only when that fails, would insure the minimum number of questions. It would also mean, however, that work might be expended searching for a subgoal, arriving perhaps at a less than definite answer, when the user already knew the answer with certainty. In response to this, some of the attributes have been labelled as LABDATA, indicating that they represent quantities which are often available as quantitative results of laboratory tests. In this case the deduce-then-ask procedure is reversed, and the system will attempt to deduce the answer only if the user cannot supply it. Given a desire to minimize both tree search and the number of questions asked, there is no guaranteed optimal solution to the problem of deciding when to ask for information, and when to try to deduce it. But the LABDATA - clinical data distinction used here has performed quite well, and seems to embody a very appropriate criterion.

Three other recent additions to the tree search procedure have helped improve performance. First, before the entire list of rules for a subgoal is retrieved, the system attempts to find a sequence of rules which would establish the goal with certainty, based only on what is currently known. Since this is a search for a sequence of rules with $CF=1$, we have termed the result a *unity path*. Besides efficiency considerations, this process offers the advantage of allowing the system to make 'common sense' deductions with a minimum of effort (rules with $CF=1$ are largely definitional). Since it also helps minimize the number of questions, this check is performed even before asking about LABDATA type attributes as well. Because there are few such rules in the system, the search is typically very brief.

Second, a straightforward bookkeeping mechanism notes the rules that have failed previously, and avoids ever trying to reevaluate any of them. (Recall that a rule may have more than one conclusion, may conclude about more than a single attribute, and hence may get retrieved more than once)

Finally, we have implemented a partial evaluation of rule premises. Since many attributes are found in several rules, the value of one clause (perhaps the last) in a premise may already have been established, even while the rest are still unknown. If this clause alone would make the premise false, there is clearly no reason to do all the search necessary to try to establish the others. Each premise is thus 'previewed' by evaluating it on the basis of currently available information. This produces a Boolean combination of TRUEs, FALSEs, and UNKNOWNs, and straightforward simplification (e.g. $F \wedge U \equiv F$) indicates whether the rule is guaranteed to fail.

This last technique is implemented in a way which suggests the utility of stylized coding in the rules. It also forms an example of what was alluded to earlier, where it was noted that the rules may be examined by various elements of the system, as well as executed. We require a way to tell if any clause in the premise is known to be false. We cannot simply EVAL each individually, since a

subgoal which had never been traced before would send the system off on its recursive search. Inhibiting search by means of some global switch will not work, since a null result could mean either that the subgoal had never been traced, or that it had been but the system was unable to deduce an answer. Even maintaining lists of previously successful or unsuccessful clauses would be an incomplete solution, since the current clause might differ only in its predicate function.

However, if we can establish which attribute is referenced by the clause, it is possible to determine (by reference to internal flags) whether it has been traced previously. If so, the clause can be EVALed to obtain the value. This is made possible by the TEMPLATE which each function name has on its property list (Figure 6).

Function	Template	Sample function call
SAME	(SAME CNTXT PARM VALUE)	(SAME CNTXT SITE BLOOD)

Figure 6.

The template indicates the generic type and order of arguments to the predicate function, much like a simplified procedure declaration. By using it as a guide, the previewing mechanism can extract the attribute from the clause, and determine whether or not it has been traced.

There are two points of interest here — first, part of the system is 'reading' the code (the rules) being executed by another part; and second, this reading is guided by the information carried in components of the rules themselves. The ability to 'read' the code could have been accomplished by requiring all predicate functions to use the same format, but this is obviously awkward. By allowing each function to describe the format of its own calls, we permit code which is stylized without being constrained to a single form, and hence is flexible and much easier to use. We require only that each form be expressible in a template built from the current set of template primitives (e.g., PARM, VALUE, etc). This approach also insures that the capability will persist in the face of future additions to the system. The result is one example of the general idea of giving the system access to, and an "understanding" of its own representations. This idea has been used and discussed extensively in [Davis1975b].

We have also implemented antecedent-style rules. These are rules which are invoked if a conclusion is made which matches their premise condition. They are currently limited to common-sense deductions (i.e. CF=1), and exist primarily to improve system efficiency. Thus, for example, if the user responds to the question of organism identity with an answer he is certain of, there is an antecedent rule which will deduce the organism gramstain and morphology. This saves the trouble of deducing these answers later via the subgoal mechanism described above.

[3.5] META-RULES

With the system's current collection of 200 rules, exhaustive invocation of rules would be quite feasible, since the maximum number of rules for a single subgoal is about 30. We are aware, however, of the problems that may occur if and when the collection grows substantially larger. It was partly in response to this that we developed an alternative to exhaustive invocation by implementing the concept of *meta-rules*. These are strategy rules which suggest the best approach to a given subgoal. They have the same format as the clinical rules (Figure 7), but can indicate that certain clinical rules should be tried first, last, before others, or not at all.

PREMISE: (\$AND (MEMBF SITE CNTXT NONSTERILESITES)
(THEREARE OBJRULES(MENTIONS CNTXT PREMISE SAMEBUG))

ACTION: (CONCLIST CNTXT UTILITY YES TALLY -1.0)

If 1) the site of the culture is one of the nonsterilesites, and
 2) there are rules which mention in their premise a previous
 organism which may be the same as the current organism

Then it is definite (1.0) that each of them is not going to be useful.

Figure 7

Thus before processing the entire list of rules applicable to any subgoal, the meta-rules for that subgoal are evaluated. They may rearrange or shorten the list, effectively ordering the search or pruning the tree. By making them specific to a given subgoal, we can specify precise heuristics without imposing any extra overhead in the tracing of other subgoals.

Note, however, that there is no reason to stop at one level of meta-rules. We can generalize this process so that, before invoking any list of rules, we check for the existence of rules of the next higher order to use in pruning or rearranging the first list. Thus, while meta-rules are strategies for selecting clinical rules, second order meta-rules would contain information about which strategy to try, third order rules would suggest criteria for deciding how to choose a strategy, etc. These higher order rules represent a search by the system through "strategy space", and appear to be powerful constraints on the search process at lower levels. (We have not yet encountered higher order meta-rules in practice, but neither have we actively sought them).

Note also that since the system's rule unwinding may be viewed as tree search, we have the appearance of a search through a tree with the interesting property that each branch point contains information on the best path to take next. Since the meta-rules can be judgmental, there exists the capability of writing numerous, perhaps conflicting heuristics, and having their combined judgment suggest the best path. Finally, since meta-rules refer to the clinical rules by their content rather than by name, the method automatically adjusts to the addition or deletion of clinical rules, as well as modifications to any of them.

The capability of meta-rules to order or prune the search tree has proved to be useful in

dealing with another variety of knowledge as well. For the sake of human engineering, for example, it makes good sense to ask the user first about the positive cultures (those showing bacterial growth), before asking about negative cultures. Formerly, this design choice was embedded in the ordering of a list buried in the system code. Yet it can be stated quite easily and explicitly in a meta-rule, yielding the significant advantages of making it both readily explainable and modifiable. Meta-rules have thus proved capable of expressing a limited subset of the knowledge formerly embedded in the control structure code of the system.

Meta-rules may also be used to control antecedent rule invocation. Thus we can write strategies which control the depth and breadth of conclusions drawn by the system in response to a new piece of information.

A detailed overview of all of these mechanisms is included in the Appendix, and indicates the way they function together to insure an efficient search for each subgoal.

The final aspect of the control structure is the tree of contexts (recall the dual meaning of the term, section [3.3]), constructed dynamically from a fixed hierarchy as the consultation proceeds (Figure 8). This serves several purposes. First, bindings of free variables in a rule are established by the context in which the rule is invoked, with the standard access to contexts which are its ancestors. Second, since this tree is intended to reflect the relationships of objects in the domain, it helps structure the consultation in ways familiar to the user. In the current domain, a patient has one or more infections, each of which may have one or more associated cultures, each of which in turn may have one or more organisms growing in it, and so on. Finally, we have found it useful to select one or more of the attributes of each context type and establish these as its MAINPROPS, or primary properties. Each time a new context of that type is sprouted, these MAINPROPS are automatically traced.⁵ Since many of them are LABDATA-type attributes, the effect is to begin each new context with a set of standard questions appropriate to that context, which serve to 'set the stage' for subsequent questions. This has proved to be a very useful human engineering feature in a domain which has evolved a heavily stylized format for the presentation of information.

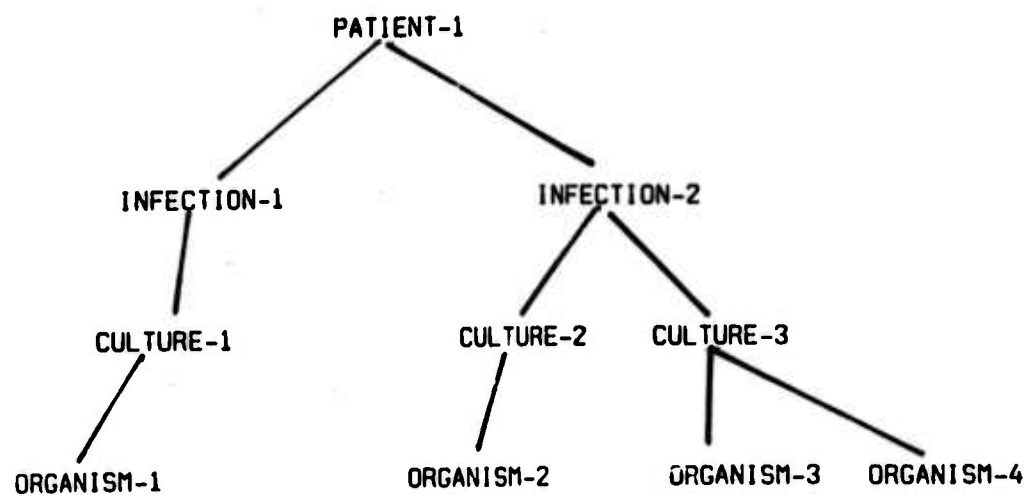


FIGURE 8

[4] RELATION TO OTHER WORK

We outline briefly in this section a few programs that relate to various aspects of our work. Some of these have provided the intellectual basis from which the present system evolved, others have employed techniques which are similar, while still others have attempted to solve closely related problems. Space limitations preclude detailed comparisons, but we indicate some of the more important distinctions and similarities.

There have been a large number of attempts to aid medical decision making (see [Shortliffe1974a] for an extensive review). The basis for some programs has been simple algorithmic processes, often implemented as decision trees ([Meyer1973], [Warner1972]), or more complex control structures in systems tailored to specific disorders [Bleich1971]. Many have based their diagnostic capabilities on variations of Bayes' theorem [Gorry1968, Warner1964], or on techniques derived from utility theory of operations research [Gorry1973a]. Models of the patient or disease process have been used successfully in [Silverman1974], [Pauker1975], and [Kulikowski1974]. A few recent efforts have been based on some form of symbolic reasoning. In particular, the glaucoma diagnosis system described in [Kulikowski1974] and the diagnosis system of [Pople1972] can also be viewed as rule-based.

Carbonell's work [Carbonell1970] represents an early attempt to make uncertain inferences in a domain of concepts that are strongly linked, much as MYCIN's are. Although the purpose of Carbonell's system was computer-aided instruction rather than consultation, much of our initial design was influenced by his semantic net model.

The basic production rule methodology has been applied in many different contexts, in attempts to solve a wide range of problems (see, for example, [Davis1975a] for an overview). The most directly relevant of these is the DENDRAL system [Buchanan1971], which has achieved a high level of performance on the task of mass spectrum analysis. Much of the initial design of MYCIN was influenced by the experience gained in building and using the DENDRAL system. Earlier attempts to build general and powerful production rule based systems have included [Waterman1970] and [Moore1973].

There have been numerous attempts to create models of inexact reasoning. Among the more recent is [LeFavire1974], which reports on the implementation of a language to facilitate fuzzy reasoning. It deals with many of the same issues of reasoning under uncertainty that are detailed in [Shortliffe1975b].

The approach to natural language used in our system has been thus far quite elementary, primarily keyword-based. Some of the work reported in [Colby1974] suggested to us initially that this might be a sufficiently powerful approach for our purposes. This has proven generally true because the technical language of this domain contains relatively few ambiguous words.

The chess playing program of [Zobrist1973] employs a knowledge representation which is

functionally quite close to ours. The knowledge base of that system consists of small sequences of code which recognize patterns of pieces, and then conclude (with a variable weighting factor) the value of obtaining that configuration. They report quite favorably on the ease of augmenting a knowledge base organized along these lines.

The natural language understanding system of [Winograd1972] had some basic explanation capabilities similar to those described here, and could discuss its actions and plans.

As we have noted above, and will explore further below, part of our work has been involved in making it possible for the system to understand its own operation. Many of the explanation capabilities were designed and implemented with this in mind, and it has significantly influenced design of the knowledge acquisition system as well. These efforts are related in a general way to the long sequence of attempts to build program-understanding systems. Such efforts have been motivated by, among other things, the desire to prove correctness of programs (as in [Waldinger1974] or [Manna1969]), and as a basis for automatic programming (as in [Green1974]). Most of these systems attempt to assign meaning to the code of some standard programming language like LISP, or ALGOL. Our attempts have been oriented toward supplying a semantics for what amounts to a high level language — the production rules and their associated components. The task is of course made easier by approaching it at this higher conceptual level — we attempt only to assign semantics to conceptual primitives of the domain like the function SAME, rather than the LISP code in which they are implemented. We cannot therefore prove that the implementation is correct, but can use the representation of meaning in other powerful ways. It forms, for example, the basis for much of the knowledge acquisition system (see section [6.3]), and permits the explanation system to be quite precise in explaining the program's actions (see [Davis1975b] for details). A similar sort of high level approach has been explored by Hewitt in his proposed INTENDER system [Hewitt1971].

Finally, similar efforts at computer-based consultants have recently been developed in different domains. The work detailed in [Nilsson1975] and [Hart1975] has explored the use of a consultation system similar to the one described here, as part of an integrated vision, manipulation, and problem solving system. Recent work on an intelligent terminal system [RAND1975] has been based in part on a formalism which grew out of early experience with the MYCIN system.

[5] FUNDAMENTAL ASSUMPTIONS OF THE METHODOLOGY

We attempt here to examine some of the explicit and implicit assumptions which are part of the methodology described above. We believe this will help to suggest the range of application for these techniques, and indicate some of their strengths and limitations. Because such a listing is potentially open-ended, we include here the assumptions essential to the methodology used in MYCIN, but not necessarily applicable to every interactive program.

There are several assumptions implicit in both the character of the rules and the ways in which they are used. First, it must be possible to write such judgmental rules. Not every domain will support this. It appears to require a field which has attained a certain level of formalization, which includes perhaps a generally recognized set of primitives and a minimal understanding of basic processes. It does not seem to extend to one which has achieved a thorough, highly formalized level, however. Assigning certainty factors to a rule should thus be a reasonable task whose results would be repeatable, but not a trivial one in which all answers were 1.

Second, we require a domain in which there is a limited sort of interaction between conceptual primitives. Our experience has suggested that a rule with more than about six clauses in the premise becomes conceptually unwieldy. The number of factors interacting in a premise to trigger an action therefore has a practical (but no theoretical) upper limit. Also, the AND/OR goal tree mechanism requires that the clauses of a rule premise can be set up as non-conflicting subgoals for the purposes of establishing each of them (just as in robot problem solving; see [Fahlman 1974] and the comment on side effects in [Siklossy 1973]). Failure of this criterion causes results which depend on the order in which evidence is collected. We are thus making fundamental assumptions concerning two forms of interaction — we assume (a) that only a small number of factors (about 6) must be considered simultaneously to trigger an action; and (b) that the presence or absence of each of those factors can be established without adverse effect on the others.

Also, certain characteristics of the domain will influence the continued utility of this approach as the knowledge base of rules grows. Where there are a limited number of attributes for a given object, the growth in the number of rules in the knowledge base will not produce an exponential growth in search time for the consultation system. Thus as newly acquired rules begin to reference only established attributes, use of these rules in a consultation will not produce further branching, since the attributes mentioned in their premises will have already been traced.

There are essential assumptions as well in the use of this methodology as the basis for an interactive system. First, our explanation capabilities (reviewed below) rest on the assumption that display of either a rule or some segment of the control flow is a reasonable explanation of system behavior. Second, much of the approach to rule acquisition is predicated on the assumption that experts can be "debriefed", that is, they can recognize and then formalize chunks of their own knowledge and experience, and express them as rules. Third, the IF/THEN format of rules must be

sufficiently simple, expressive, and intuitive that it can provide a useful language for expressing such formalizations. Finally, the system's mode of reasoning (a simple *modus ponens* chaining) must appear natural enough that a user can readily follow along. We offer below (section [6]) arguments that all these are plausible assumptions.

There is an important assumption, too, in the development of a system for use by two classes of users. Since the domain experts who educate the system so strongly influence its conceptual primitives, vocabulary, and knowledge base, we must be sure that the naive users who come for advice speak the same language.

The approach we describe does not, therefore, seem well suited to domains requiring a great deal of complex interaction between goals, or those for which it is difficult to compose sound judgmental rules. As a general indication of potentially useful applications, we have found that cognitive tasks are good candidates. In one such domain, antibiotic therapy selection, we have met with encouraging success.

[6] PRODUCTION RULES AS A KNOWLEDGE REPRESENTATION

In the introduction to this report we outlined three design goals for the system we are developing: usefulness (including competence), maintenance of an evolutionary knowledge base, and support of an interactive consultation. Our experience has suggested that production rules offer a knowledge representation that greatly facilitates the accomplishment of these goals. Such rules are straightforward enough to make feasible many interesting features beyond performance, yet powerful enough to supply significant problem solving capabilities. Among the features discussed below are the ability for explanation of system performance, and acquisition of new rules, as well as the general 'understanding' by the system of its own knowledge base.

[6.1] COMPETENCE

[6.1.1] *Advantages of production rules*

Recent problem solving efforts in AI have made it clear that high performance of a system is often strongly correlated with the depth and breadth of the knowledge base. Hence, the task of accumulation and management of a large and evolving knowledge base soon poses problems which dominate those encountered in the initial phases of knowledge base construction. Our experience suggests that giving the system itself the ability to examine and manipulate its knowledge base provides some capabilities for confronting these problems. These are discussed in subsequent sections below.

The selection of production rules as a knowledge representation is in part a response to this fact. One view of a production rule is as a modular segment of code [Winograd 1975], which is heavily stylized [Waterman 1970, Buchanan 1971]. Each of MYCIN's rules is, as noted, a simple conditional statement: the premise is constrained to be a Boolean expression, the action contains one or more conclusions, and each is completely modular and independent of the others. Such *modular, stylized coding* is an important factor in building a system that is to achieve a high level of competence.

For example, any stylized code is easier to examine. This is used in several ways in the system. Initial integration of new rules into the knowledge base can be automated, since their premise and action parts can be systematically scanned, and the rules can then be added to the appropriate internal lists. In the question answering system, inquiries of the form 'Do you recommend clindamycin for bacteroides?' can be answered by retrieving rules whose premise and action contain the relevant items. Similarly, the detection of straightforward cases of contradiction and subsumption is made possible by the ability to examine rule contents. Stylized code also makes feasible the direct manipulation of individual rules, facilitating automatic correction of such undesirable interactions. The discussion below of the use of TEMPLATES demonstrates yet another use, in which this

capability is used to help provide explanations of system behavior.

The benefits of modularized code are well understood. Especially significant in this case are the ease of adding new rules and the relatively uncomplicated control structure which the modular rules permit. Since rules are retrieved because they are relevant to a specific goal (i.e., they mention that goal in their action part), the addition of a new rule requires only that it be added to the appropriate internal list according to the clinical parameters found in its action. A straightforward depth first search (the result of the backward chaining of rules) is made possible by the lack of interactions among rules.

These benefits are common to stylized code of any form. Stylization in the form of production rules in particular has proved to be a useful formalism for several reasons. In the domain of deductive problems, especially, it has proven to be a natural way of expressing knowledge. It also supplies a clear and convenient way of expressing modular chunks of knowledge, since all necessary context is stated explicitly in the premise. This in turn makes it easier to insure proper retrieval and use of each rule. Finally, in common with similar formalisms, one rule never directly calls another. This is a significant advantage in integrating a new rule into the system - it can simply be 'added to the pot' and no other rule need be changed to insure that it is called (compare this with the addition of a new procedure to a typical ALGOL-type program).

[6.1.2] *Shortcomings of production rules*

Stylization and modularity also result in certain shortcomings, however. It is, of course, somewhat harder to express a given piece of knowledge if it must be put into a predetermined format. The intent of a few of the rules in our system are thus less than obvious to the naive user even when translated into English. The requirement of modularity (along with the uniformity of the knowledge base), means all necessary contextual information must be stated explicitly in the premise, and this at times leads to rules which have awkwardly long and complicated premises.

Shortcomings in the formalism arise also in part from the backward chaining control structure, and from the restriction to a pure predicate in the premise. It is not always easy to map a sequence of desired actions or tests into a set of production rules whose goal-directed invocation will provide that sequence. Thus, while the system's performance is reassuringly similar to some human reasoning behavior, the creation of appropriate rules which result in such behavior is at times non-trivial. This may in fact be due more to programming experience oriented primarily toward ALGOL-like languages, rather than any essential characteristic of production rules. After some experience with the system we have improved our skill at 'thinking backward'.

A final shortcoming arises from constraining rule premises to be simple predicates. This forces a pure problem reduction mode in the use of rules: each clause of a premise is set up as an independent goal, and execution of the action should be dependent solely on the success or failure of premise evaluation, without referencing its precise value. It is at times, however, extremely convenient to write what amounts to a 'for each' rule, as in 'for each organism such that ... conclude ...'. A few rules of this form are present in the system (including, for example, the meta-rule in

Figure 7), and they are made to appear formally like the rest by allowing the premise to compute a value (the set of items that satisfy the premise), which is passed to the action clause via a global variable. While this has been relatively successful, the violation of the basic formalism results in other difficulties — in particular, in the explanation system, which produces somewhat murky explanations of such rules. We are working toward a cleaner solution of this problem.

[6.2] EXPLANATION

Augmentation or modification of any knowledge base is facilitated by the ability to discover what knowledge is currently in the system and how it is used. The system's acceptance (especially to a medical audience) will be strongly dependent upon the extent to which its performance is natural (i.e., human-like) and transparent. Lack of acceptance of some applications programs can be traced to their obscure reasoning mechanisms which leave the user forced to accept or reject advice without a chance to discover its basis. One of our original design criteria, then, was to give the system the ability to provide explanations of its behavior and knowledge. It soon became evident that an approach relying on some form of symbolic reasoning (rather than, for example, statistics) would make this feasible. This was one of the primary reasons behind the choice of the production rule representation, and has continued to influence the program's development.

Our initial efforts at explanation and question-answering were based on three capabilities: (i) display on demand during the consultation the rule currently being invoked, (ii) record rules which were invoked, and after the consultation, be able to associate specific rules with specific events (questions and conclusions) to explain why each of them happened, and (iii) search the knowledge base for a specific type of rule in answer to inquiries of the user. The first of these could be easily implemented via the single-word command format described below.

The latter two were intended for use after the consultation, and hence were provided with a simple natural language front end. Examples are shown in Figure 9 (additional examples can be found in [Shortliffe1975a]). Note that the capability for answering questions of type (ii) has been extended to include inquiries about actions the program *failed* to take (example [d], Figure 9). This is based on the ability of the explanation system to simulate the control structure of the consultation system, and can be extremely useful in deciphering the program's behavior. For questions of type (iii), (e) in Figure 9) the search through the knowledge base is directed by a simple parsing of the question into a request for a set of rules, with constraints on premise and/or action contents. The retrieval of relevant rules is guided primarily by pre-established (but automatically generated) lists which indicate premise and action contents.

<insert figure 9 about here>

Some generalization of and extensions to the methodology of (i) and (ii) have been motivated by two shortcomings. Displaying the current rule is not particularly informative if the rule is essentially definitional and hence conceptually trivial. The problem here is the lack of a good gauge for the amount of information in a rule.

Recording individual rule invocations, questions, and conclusions is useful, but, as a record of individual events, it fails to capture the context and ongoing sequence. It is difficult therefore to explain any event with reference to anything but the specific information recorded with that event.

Two related techniques were developed to solve these problems. First, to provide a metric for the amount of information in a rule, we use (in a very rough analogy with information theory) $-(\log CF)$. Rules which are definitional ($CF = 1$) have by this measure no information, while those

(a) ** HOW DID YOU DECIDE THAT ORGANISM-1 MIGHT BE AN E.COLI?

I used RULE021 to conclude that the identity of ORGANISM-1 is e.coli. This gave a cumulative CF of (.47).

Then I used RULE084 to conclude that the identity of ORGANISM-1 is e.coli. This gave a cumulative CF of (.55).

Finally, I used RULE003 to conclude that the identity of ORGANISM-1 is e.coli. This gave a cumulative CF of (.74).

(b) ** DID YOU USE RULE 163 TO FIND OUT ANYTHING ABOUT ORGANISM-1?

RULE163 was tried in the context of ORGANISM-1, but it failed because it is not true that the patient has had a genito-urinary tract manipulative procedure (clause 3).

(c) ** DID RULE 116 TELL YOU ANYTHING ABOUT INFECTION-1?

No. RULE116 would be relevant to the primary-bacteremia when applied to ORGANISM-1.

But clause 2 ("the gram stain of the organism is not known") was already known to be false for ORGANISM-1, so the rule was never tried.

(d) ** WHY DIDN'T YOU CONSIDER STREPTOCOCCUS AS A POSSIBILITY FOR ORGANISM-1?

The following rule could have been used to determine that the identity of ORGANISM-1 was streptococcus: RULE033

But clause 2 ("the morphology of the organism is coccus") was already known to be false for ORGANISM-1, so the rule was never tried.

(e) ** DO YOU RECOMMEND CLINDAMYCIN FOR BACTEROIDES?

Yes.

RULE060

If the identity of the organism is bacteroides

Then I recommend therapy be chosen from among the following:

CHLORAMPHENICOL, CLINDAMYCIN, RIFAMPIN, ERYTHROMYCIN, CARBENICILLIN
TETRACYCLINE, LINCOMYCIN

(f) ** IS BLOOD A STERILE SITE?

Yes, blood is one of those sites that are normally sterile.

(g) ** WHAT ARE THE ORGANISMS ASSOCIATED WITH THE GASTRO-INTESTINAL TRACT?

The likely pathogens associated with the gastro-intestinal tract are:
ENTEROCOCCUS, CLOSTRIDIUM-GANGRENE, BACTEROIDES, KLEBSIELLA, PSEUDOMONAS,
E.COLI, ENTEROBACTER, PROTEUS.

Figure 9

which express less obvious implications have progressively more information. The measure is clearly imperfect, since first, CFs are not probabilities, and there is thus no formal justification that $-(\log CF)$ is a meaningful measure. Second, any sophisticated information content measure should factor in the state of the observer's knowledge, since the best explanations are those which are based on an understanding of what the observer fails to comprehend. Despite these shortcomings, however, this heuristic has proven to be quite useful.

To solve the second problem (explaining events in context) the process of recording individual rule invocations has been generalized: all the basic control functions of the system have been augmented to leave behind a history of each of their executions. This internal trace is then read by various parts of the explanation system to provide a complete, in-context explanation of any part of the system's behavior.

Because the consultation process is essentially one of search through an AND/OR goal tree, inquiries during the course of a consultation fall quite naturally into two types: WHY a question was asked, and HOW a conclusion was (or will be) reached. The first of these looks "up" the tree, in examining higher goals, while the second looks "down" in examining rules which may help achieve a goal. This part of the system's explanation capability can thus be viewed in general as a process of tree traversal. By combining this concept with the information content metric, we make possible explanations in varying levels of detail: the tree may be traversed in steps whose information content size is specified by the user, rather than simply stepping from goal to goal. At the start of the traversal process, "information distance" from the current goal to the top of the tree is normalized to 9. The argument to the WHY command (an integer between one and ten, assumed to be one if absent) is then taken to indicate some part of that distance. Thus, WHY 3 indicates an explanation which encompasses approximately one-third of the total "conceptual" distance. Repeated WHY questions have the effect of stepping up the goal tree. Examples are shown below; additional examples are found in [Shortliffe1975a]).

In this series, how many of the cultures were positive for the
e.coli (ORGANISM-1)? {typical system prompt}
**WHY

{request from user to determine motivation for question}

[i.e. WHY is is important to determine the number of cultures which were
positive for the organism?]

{system's rephrasing of the WHY question, and answer}

[1.0] This will help to determine whether there is significant disease
associated with this occurrence of the organism.

It has already been established that

[1.1] the culture was taken from a sterile source

[1.2] the number of cultures in the series is greater than 2

Therefore, if

[1.3] the number of cultures which were positive for ORGANISM-1
is greater than 2

Then there is suggestive evidence (.5) that there is significant disease
associated with this occurrence of the e.coli (ORGANISM-1).

[RULE049]

WHY *{repeated WHY to determine relevance of previous explanation}*
 [i.e. WHY is it important to determine if there is significant disease associated with this occurrence of the organism?]
 [2.0] This will help to determine a diagnosis for which the infection should be treated.
 It has already been established that
 [2.1] the infection is not one of: primary-bacteremia, meningitis
 [2.2] for some current culture of the patient it is true that the site of the culture is blood
 Therefore, if
 [2.3] there is significant disease associated with this occurrence of the organism
 Then
 it is definite (1.0) that the diagnosis for which the infection should be treated is secondary-bacteremia
 (RULE103)

Figure 10

Similarly, HOW commands step "down" the tree. This can result in examining either branches which have already been traversed (as in the example below) or those which have yet to be tried (in which case the question becomes 'HOW will you determine...')

****HOW 1.1** *{the 11 refers to the information in the line labelled /11/ in Fig 10}*
 [i.e. HOW was it determined that the culture was taken from a sterile source?]
 [3.1] RULE001 was used to conclude that there is suggestive evidence (.7) that the culture was taken from a sterile source.
 [3.2] RULE022 was used to conclude that there is strongly suggestive evidence (.7) that the culture was taken from a sterile source.
****HOW 3.1** *{another request from the user}*
 [i.e. HOW was RULE001 used?]
 It has already been established that
 [4.1] the site of the culture is one of: those sites from which the sterility of cultures depends upon the method of collection
 [4.2] the method of collection of the culture is one of: the recognized methods of collection associated with the site of the culture, and
 [4.3] it is not known whether care was taken in collecting the culture
 Therefore
 there is strongly suggestive evidence (.8) that the culture was taken from a sterile source
 (RULE022)

Figure 11

The system's fundamental approach to explanation is thus to display some recap of its internal

actions, a trace of its reasoning. The success of this technique is predicated on the claim that the system's basic approach to the problem is sufficiently intuitive that a summary of those actions is at least a reasonable basis from which to start. While it would be difficult to prove the claim in any formal sense, there are several factors which suggest its plausibility.

First, we are dealing with a domain in which deduction, and deduction in the face of uncertainty, is a primary task. The use of production rules in an IF/THEN format seems therefore to be a natural way of expressing things about the domain, and the display of such rules should be comprehensible. Second, the use of such rules in a backward chaining mode is, we claim, a reasonably intuitive scheme. *Modus ponens* is a well-understood and widely (if not explicitly) used mode of inference. Thus, the general form of the representation and the way it is employed should not be unfamiliar to the average user. More specifically, however, consider the source of the rules. They have been given to us by human experts who were attempting to formalize their own knowledge of the domain. As such, they embody accepted patterns of human reasoning, implying that they should be relatively easy to understand, especially for those familiar with the domain. As such, they will also attack the problem at what has been judged an appropriate level of detail. That is, they will embody the right size of "chunks" of the problem to be comprehensible.

We are not, therefore, recapping the binary bit level operations of the machine instructions for an obscure piece of code. We claim instead to be working with primitives and a methodology whose (a) substance, (b) level of detail, and (c) mechanism are all well suited to the domain, and to human comprehension, precisely because they were provided by human experts. This approach seems to provide what may plausibly be an understandable explanation of system behavior.

This use of symbolic reasoning is one factor which makes the generation of explanations an easier task. For example, it makes the display of a backtrace of performance comprehensible (as, for example, in Figure 9). The basic control structure of the consultation system is a second factor. The simple depth-first-search of the AND/OR goal tree makes HOW, WHY, and the tree traversal approach natural (as in Figures 10 and 11). We believe several concepts in the current system are, however, fairly general purpose, and would be useful even in systems which did not share these advantages. Whatever control structure is employed, the maintenance of an internal trace will clearly be useful in subsequent explanations of system behavior. The use of some information metric will help to insure that those explanations are at an appropriate level of detail. Finally, the explanation generating routines require some ability to decipher the actions of the main system.

Three different means of generating explanations are present in our current system. (a) The first type is used in producing answers to HOW questions which explore branches of the tree not yet traversed by the consultation system (e.g. "HOW will you determine the identity of the organism?"). These are produced by having the explanation system simulate the operation of the consultation system via special purpose software. This is thus a hand-crafted solution. (b) More general is the use of the goal-tree concept as a basis for explanation — since the notion of an AND/OR goal tree models a large part of the control structure, we have a single, uncomplicated models for much of the system's behavior. As a result, a relatively simple formalism which equates

WHY and HOW with tree traversal offers a reasonably powerful and comprehensive explanatory capability. (c) The most general technique has been mentioned earlier — one part of the system "reads" the code (the rule) that is being executed by another part (The same basic approach is used to do the partial evaluation described above and the division of rule premises into known and unknown clauses shown in the examples above). In this case the explanations are constructed by reference to the contents of the rules, and this referral is guided by information contained in the rule components themselves.

By way of contrast, we might try to imagine how a program based on a statistical approach might attempt to explain itself. Such systems can, for instance, display a disease which has been deduced and a list of relevant symptoms, with prior and posterior probabilities. No more informative detail is available, however. When the symptom list is long, it may not be clear how each of them (or some combination of them) contributed to the conclusion. It is more difficult to imagine what sort of explanation could be provided if the program were interrupted with interim queries while in the process of computing probabilities. The problem, of course, is that statistical methods are not good models of the actual reasoning process (as shown in psychological experiments of [Edwards1968] and [Tversky1974]), nor were they designed to be. While they are operationally effective when extensive data concerning disease incidence are available, they are also for the most part, "shallow", one-step techniques which capture little of the ongoing process actually used by expert problem solvers in the domain.⁶

We have found the presence of even the current basic explanation capabilities to be extremely useful, as they have begun to pass the most fundamental test: it has become easier to ask the system what it did than to trace through the code by hand. The continued development and generalization of these capabilities is one focus of our present research.

[6.3] ACQUISITION

Since the field of infectious disease therapy is both large and constantly changing, it was apparent from the outset that the program would have to deal with an evolving knowledge base. The domain size made writing a complete set of rules an impossible task, so the system was designed to facilitate an incremental approach to competence. New research in the domain produces new results and modifications of old principles, so that a broad scope of knowledge-base management capabilities was clearly necessary.

As suggested above, a fundamental assumption is that the expert teaching the system can be "debriefed", thus transferring his knowledge to the program. That is, presented with any conclusion he makes during a consultation, the expert must be able to state a rule indicating all relevant premises for that conclusion. The rule must, in and of itself, represent a valid chunk of clinical knowledge.

There are two reasons why this seems a plausible approach to knowledge acquisition. First, clinical medicine appears to be at the correct level of formalization. That is, while relatively little of the knowledge can be specified in precise algorithms (at a level comparable to, say, elementary physics) the judgmental knowledge that exists is often specifiable in reasonably firm heuristics. Second, on the model of a medical student's clinical training, we have emphasized the acquisition of new knowledge in the context of debugging (although the system is prepared to accept a new rule from the user at any time). We expect that some error on the system's part will become apparent during the consultation, perhaps through an incorrect organism identification or therapy selection. Tracking down this error by tracing back through the program's actions is a reasonably straightforward process which presents the expert with a methodical and complete review of the system's reasoning. He is obligated to either approve of each step or to correct it. This means that the expert is faced with a sharply focussed task of adding a chunk of knowledge to remedy a specific bug. This makes it far easier for him to formalize his knowledge than would be the case if he were asked, for example, "tell me about bacteremia."

This methodology has the interesting advantage that the context of the error (i.e., which conclusion was in error, what rules were used, what the facts of this case were, etc.) is of great help to the acquisition system in interpreting the expert's subsequent instructions for fixing the bug. The error type and context supply the system with a set of expectations about the form and content of the anticipated correction, and this greatly facilitates the acquisition process (details of this and much of the operation of the acquisition system are found in [Davis1975b]).

The problem of educating the system can be usefully broken down into three phases: uncovering the bug, transferring to the system the knowledge necessary to correct the bug, and integrating the new (or revised) knowledge into the knowledge base. As suggested above, the explanation system is designed to facilitate the first task by making it easy to review all of the program's actions. Corrections are then specified by adding new rules (and perhaps new values, attributes, or contexts), or by modifying old ones. This process is carried out in a mixed initiative dialogue using a subset of standard English (an early example is found in [Shortliffe1975a]).

The system's understanding of the dialog is based on what may be viewed as a primitive form of 'model-directed' automatic programming. Given some natural language text describing one clause of a new rule's premise, the system scans the text to find keywords suggesting which predicate function(s) are the most appropriate translations of the predicate(s) used in the clause. The appropriate TEMPLATE for each such function is retrieved, and the 'parsing' of the remainder of the text is guided by the attempt to fill this in.

If one of the functions were SAME, the TEMPLATE would be as shown in Figure 6. CNTXT is known to be a literal which should be left as is, PARM signifies a clinical parameter (attribute), and VALUE denotes a corresponding value. Thus the phrase "the stain of the organism is negative" would be analyzed as follows: the word "stain" in the system dictionary has as part of its semantic indicators the information that it may be used in talking about the attribute *gramstain* of an organism. The word "negative" is known to be a valid value of *gramstain* (although it has other associations as well). Thus one possible (and in fact the correct) parse is (SAME CNTXT GRAM GRAMNEG), or "the gramstain of the organism is gramnegative."

Note that this is another example of the use of higher level primitives to do a form of program understanding. It is the semantics of PARM and VALUE which guide the parse after the TEMPLATE is retrieved, and the semantics of the *gramstain* concept which allow us to insure the consistency of each parse. Thus by treating such concepts as conceptual primitives, and providing semantics at this level, we make possible the capabilities shown, using relatively modest amounts of machinery.

Other, incorrect parses are of course possible, and are generated too. There are three factors, however, which keep the total number of parses within reasonable bounds. First, and perhaps most important, we are dealing with a very small amount of text. The user is prompted for each clause of the premise individually, and while he may type an arbitrary amount at each prompt, the typical response is less than a dozen words. Second, there is a relatively small degree of ambiguity in the semi-formal language of medicine. Therefore a keyword-based approach produces only a small number of possible interpretations for each word. Finally, insuring the consistency of any given parse (eg. that VALUE is indeed a valid value for PARM) further restricts the total number generated. Typically, between 1 and 15 candidate parses result.

Ranking of possible interpretations of a clause depends on expectation and internal consistency. As noted above, the context of the original error supplies expectations about the form of the new rule, and this is used to help sort the resulting parses to choose the most likely.

As the last step in educating the system, we have to integrate the new knowledge into the rest of the knowledge base. We have only recently begun work on this problem, but we recognize two important, general problems. First, the rule set should be free of internal contradictions, subsumptions, or redundancies. The issue is complicated significantly by the judgmental nature of the rules. While some inconsistencies are immediately obvious (two rules identical except for differing certainty factors) indirect contradictions, (resulting from chaining rules, for example) are more difficult to detect. Inexactness in the rules means that we can specify only an interval of

consistent values for a certainty factor.

The second problem is coping with the secondary effects that the addition of new knowledge typically introduces. This arises primarily from the acquisition of a new value, clinical parameter or context. After requesting the information required to specify the new structure, it is often necessary to update several other information structures in the system, and these in turn may cause yet other updating to occur. For example, the creation of a new value for the site of a culture involves a long sequence of actions: the new site must be added to the internal list ALLSITES, it must then be classified as either sterile or non-sterile, and then added to the appropriate list; if non-sterile the user has to supply the names of the organisms that are typically found there, and so forth. While some of this updating is apparent from the structures themselves, much of it is not. We are currently investigating methods for specifying such interactions, and a methodology of representation design that minimizes or simplifies the interactions to begin with.

The choice of a production rule representation does impose some limitations in the knowledge transfer task. Since rules are simple conditional statements, they can at times provide power insufficient to express some more complex concepts. In addition, while expressing a single fact is often convenient, expressing a larger concept via several rules is at times somewhat more difficult. As suggested above, mapping from a sequence of actions to a set of rules is not always easy. Goal-directed chaining is apparently not currently a common human approach to structuring larger chunks of knowledge.

Despite these drawbacks, we have found the production rule formalism a powerful one. It has helped to organize and build, in a relatively short period, a knowledge base which performs at an encouraging level of competence. The rules are, as noted, a reasonably intuitive way of expressing simple chunks of inferential knowledge, and one which requires no acquaintance with any programming language. While it may not be immediately obvious how to restate domain knowledge in production rule format, we have found that infectious disease experts soon acquired some proficiency in doing this with relatively little experience.

The rules also appear capable of embodying appropriate-sized chunks of knowledge, and of expressing concepts that are significant statements. They remain, however, straightforward enough to be built of relatively simple compositions of conceptual primitives (the attributes, values, etc.) While any heavily stylized form of coding of course makes it easier to produce code, stylizing in the form of production rules in particular also provides a framework which is structurally simple enough to be translatable to simple English. This means that the experts can easily comprehend the program's explanation of what it knows, and equally easily specify knowledge to be added.

[7] CONCLUSIONS

The MYCIN system has begun to approach its design goals of competence and high performance, ⁷ flexibility in accomodating a large and changing knowledge base, and ability to explain its own reasoning. Successful applications of our control structure with rules applicable to other problem areas have been (a) fault diagnosis and repair reccommendations for bugs in an automobile horn system [van Melle1974], (b) a consultation system for industrial assembly problems [Hart1975], and (c) part of the basis for an intelligent terminal system [RAND1975].

A large factor in this work has been the production rule methodology. It has proved to be a powerful, yet flexible representation for encoding knowledge, and has contributed significantly to the capabilities of the system.

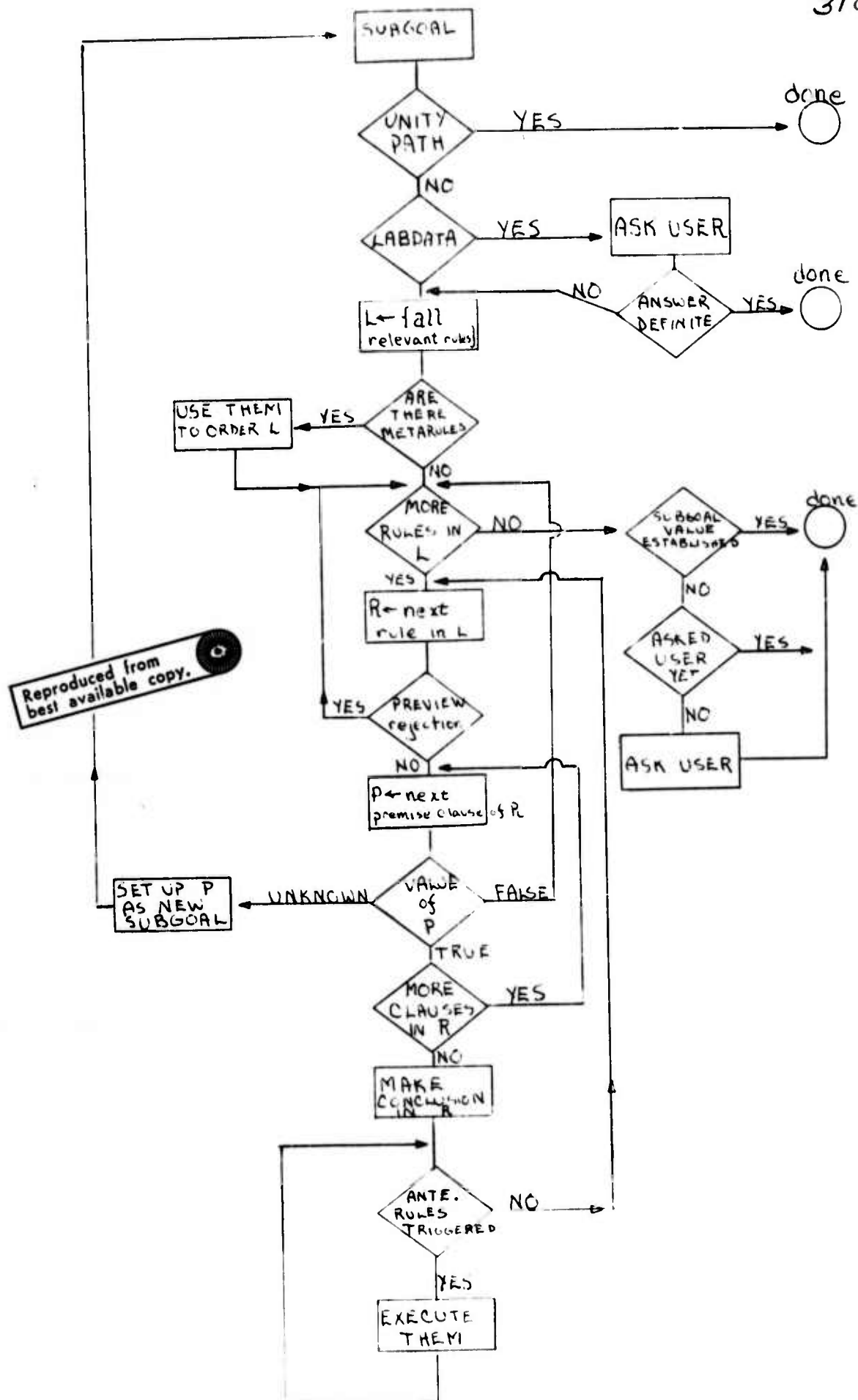


FIG 12: OVERVIEW OF CONTROL STRUCTURE

Captions

- 1: [consultation, initial segment]
Initial segment of a session with the consultation system. User responses are in **boldface** and follow the double asterisks. Italicized comments at the right are not part of the actual dialog.
- 2: [consultation, final segment]
Final segment of a consultation. Alternative choices for therapies are generated if requested.
- 3: [five boxes - 4 systems plus rule base]
The five components of the system: four programs and a single knowledge base. Arrows indicate the direction of information flow.
- 4: [rule]
A rule from the knowledge base. \$AND and \$OR are the multivalued analogues of the standard Boolean AND and OR.
- 5: [quadruples from data base]
Samples of information in the data base during a consultation.
- 6: [template]
PARM is shorthand for clinical parameter (attribute); VALUE is the corresponding value; CNTXT is a free variable which references the context in which the rule is invoked.
- 7: [meta-rule]
A meta-rule. A previous infection which has been cured (temporarily) may reoccur. Thus one of the ways to deduce the identity of the current organism is by reference to previous infections. However, this method is not valid if the current infection was cultured from one of the non-sterile culture sites. Thus this metarule says, in effect, *if the current culture is from a non-sterile site, don't bother trying to deduce the current organism identity from identities of previous organisms.*
- 8: [context tree]
A sample of the contexts which may be sprouted during a consultation.
- 9: [QA natural language examples]
Examples of natural language question answering capabilities. Questions [a]-[d] reference a specific consultation, while [e]-[g] are general inquiries answered from the system knowledge base.
- 10: [Explanation examples, WHY]
Examples of explanation capabilities. User input is in **boldface** and follows the double asterisk. Expansion of each WHY or HOW question (enclosed in brackets) is produced by the system, to be sure the user is aware of the system's interpretation of them.
- 11: [Explanation examples, HOW]
{No caption necessary.}

Appendix caption

A detailed overview of the control structure, illustrating the combination of the various mechanisms used to establish a subgoal.

NOTES

- [1] The MYCIN system has been developed by the authors in collaboration with:
Drs. Stanley Cohen, Stanton Axline, Frank Rhame, Robert Illa, and Rudolpho Chavez-Pardo, all of whom provided medical expertise;
William van Melle, who made extensive revisions to the system code for efficiency and to introduce new features;
Carlisle Scott, who (with William Clancey) designed and implemented the expanded natural language question answering capabilities.
- [2] We have recently begun investigating extending the system. The next medical domain will be the diagnosis and treatment of meningitis infections. This area is sufficiently different to be challenging, and yet similar enough to suggest that some of the automated procedures we have developed may be quite useful.
The paper by [van Melle1974] reports on an interesting effort at inserting an entirely different knowledge base into the body of the current system. A small part of an automobile repair manual was translated into production rules, and the appropriate attributes, objects, contexts, and vocabulary were provided. It then required relatively little effort to plug this new knowledge base into the standard system code, and a small but completely functional automobile consultant program resulted.
- [3] We are presently working on an interface which will allow the physician to describe the important facts of the case using an appropriate subset of natural language. This would allow him to begin by describing the case in standard medical format, and allow the system to prompt with more general questions like 'Please describe the second blood culture.'
- [4] Note that, unlike standard probability theory, $\&$ AND does not involve any multiplication over its arguments. Since CFs are not probabilities, there is no *a priori* reason why a product should be a reasonable number. There is, moreover, a long-standing convention in work with multi-valued logics which interprets AND as *min* and OR as *max* [Lukasiewicz]. It is based primarily on intuitive grounds: if a conclusion requires all of its antecedents to be true, then it is a relatively conservative strategy to use the smallest of the antecedent values as the value of the premise. Similarly, if any one of the antecedent clauses justifies the conclusion, we are safe in taking the maximum value.
- [5] As a result of this, the control flow is actually slightly more complicated than a pure AND/OR goal tree, and the flowchart in the appendix is correspondingly more complex.
- [6] However, the reasoning process of human experts may not be the ideal model for *all* knowledge-based problem solving systems. In the presence of reliable statistical data, programs using a decision theoretic approach are capable of performance surpassing those of their human counterparts.
In domains like infectious disease therapy selection, however, which are characterized by 'judgmental knowledge', statistical approaches may not be viable. This appears to be the case for many medical decision making areas. See [Gorry1973b] for further discussion of this point.
- [7] A preliminary evaluation of the system [Shortliffe1974a] demonstrates agreement with a panel of experts on 73% of a randomly chosen set of patients. A more formal evaluation study is currently being performed.

References

- [Bleich1971]
Bleich H L. The computer as a consultant *The New England Journal of Medicine*, 284 (21 January 1971) pp 141-147.
- [Buchanan1971]
Buchanan B G, Lederberg J. The heuristic *DENDRAL* program for explaining empirical data, IFIP, 1971, pp 179 - 188.
- [Colby1974]
Colby K M, Parkinson R C, Faught B L. Pattern matching rules for the recognition of natural language dialog expressions, AI Memo #234, June 1974, Computer Science Department, Stanford University.
- [Carbonell1970]
Carbonell J R. Mixed initiative man-computer instructional dialogues, BBN Report 1971, (31 May 1970), Bolt, Beranek and Newman, Cambridge, Mass.
- [Davis1975a]
Davis R, King J. An overview of production systems, in *Machine Representations of Knowledge*, Proceedings of the NATO Advanced Study Institute, Santa Cruz, CA, (in press).
- [Davis1975b]
Davis R. Forthcoming PhD thesis, Stanford University.
- [Edwards1968]
Edwards R. Conservatism in human information processing, in Kleinmuntz (Ed.), *Formal Representation of Human Judgment*, Wiley (1968), pp 17-52.
- [Fahlman1974]
Fahlman S E. A planning system for robot construction tasks, *Artificial Intelligence* 5 (Spring 1974) pp 1-50.
- [Finkel1974]
Finkel R, Taylor R, Bolles R, Paul R, Feldman J. AL, A programming system for automation, AI Memo 243, Nov 1974, Stanford University.
- [Gorry1968]
Gorry G A, Barnett G O. Experience with a model of sequential diagnosis, *Computers and Biomedical Research*, 1 (1968) pp 490-507.
- [Gorry1973a]
Gorry G A, Kassirer J P, Essig A, Schwartz W B. Decision analysis as the basis for computer-aided management of acute renal failure, *American Journal of Medicine*, 55 (1973) pp 473-484.
- [Gorry1973b]
Gorry G A. Computer-assisted clinical decision making, *Meth. Inform. Med.*, 12 (1973), pp 42-51.

[Green 1974]

Green C. C., Waldinger R. J., Barstow D. R., Elschlager P., Lenat D. B., McCune B. P., Shaw D. E., Steinberg L. I. Progress report on program-understanding systems, A I Memo #240, August 1974, Computer Science Department, Stanford University.

[Hart 1975]

Hart P. E. Progress on a computer-based consultant, AI Group Technical Note 99, January 1975, Stanford Research Institute, Menlo Park, CA.

[Hewitt 1971]

Hewitt C. Procedure semantics - models of procedures and the teaching of procedures, *Natural Language Processing*, Courant Computer Science Symposium, Rustin R. (Ed), 8 (1971) pp 331-350.

[Kagan 1973]

Kagan B. M., Fannin S. L., Bardie F. Spotlight on antimicrobial agents - 1973, *JAMA*, 226, #3 (October 1973) pp 306-310.

[Kulikowski 1974]

Kulikowski C. A. A system for computer-based medical consultation, *Proceedings of National Computer Conference*, 1974.

[LeFaivre 1975]

Le Faivre R. A. Fuzzy problem solving, Tech Rept 37 (September 1974), University of Wisconsin - Madison.

[Lesser 1974]

Lesser V. R., Fennell R. D., Erman L. D., Reddy D. R. Organization of the HEARSAYII speech understanding system, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, February 1975, pp 11-23.

[Lukasiewicz 1970]

Lukasiewicz J. A numerical interpretation of the theory of propositions, in *Jan Lukasiewicz: Selected Works* Borkowski (ed.), 1970.

[MACSYMA 1974]

The MACSYMA reference manual, September 1974, The MATHLAB Group, Mass Inst Tech.

[Manna 1969]

Manna Z. Correctness of programs, *Journal of Computer Systems Sciences*, May 1969.

[Meyer 1973]

Meyer A. V., Weissman W. K. Computer analysis of the clinical neurological exam, *Computers and Biomedical Research*, 3(1973) pp 111-117.

[Moore 1973]

Moore J., Newell A. How can MERLIN understand, Tech Rept (15 November 1973), Carnegie Mellon University.

[Nilsson1975]

Nilsson N J (Ed.) Artificial Intelligence - Research and Applications, A I Group Progress Report, May 1975, Stanford Research Institute, Menlo Park, CA.

[Pauker1975]

Pauker S G, Gorry G A, Kassirer J P, Schwartz W B. Towards the simulation of clinical cognition: taking the present illness by computer. *American Journal of Medicine*, (in press) 1975.

[Pople1972]

Pople H, Meyers J, Miller R. DIALOG, a model of diagnostic logic for internal medicine *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, USSR, September 1975, pp 848-855.

[Rand1975]

RAND intelligent terminals agent: design philosophy, RAND Report 1309, October 1975, The RAND Corporation, Santa Monica, CA.

[Rosen1975]

Rosen C A, Nitzan D. Some developments in programmable automation, AI Group Technical Note 100, January 1975, Stanford Research Institute, Menlo Park, CA.

[Shortliffe1974a]

Shortliffe E H. MYCIN- A rule-based computer program for advising physicians regarding antimicrobial therapy selection, A I Memo #251, October 1974, Computer Science Department, Stanford University. Also to appear as *MYCIN: Computer-Based Medical Consultations*, American Elsevier, 1976.

[Shortliffe1974b]

Shortliffe E H, Axline S G, Buchanan B G, Cohen, S N. Design considerations for a program to provide consultations in clinical therapeutics Presented at San Diego Biomedical Symposium 1974 (February 6-8, 1974).

[Shortliffe1975a]

Shortliffe E H, Davis R, Buchanan B G, Axline S G, Green C C, Cohen S N. Computer-based consultations in clinical therapeutics - explanation and rule acquisition capabilities of the MYCIN system, *Computers and Biomedical Research*, 8 (1975) pp 303-320.

[Shortliffe1975b]

Shortliffe E H, Buchanan B G. A model of inexact reasoning in medicine, *Mathematical Biosciences* 23 (1975) pp 351-379.

[Siklossy1973]

Siklossy L, Roach J. Proving the impossible is impossible is possible, *Proc Third Int Joint Conf AI*, 1973, Stanford University, pp 383-387.

[Silverman1974]

Silverman H. A digitalis therapy advisor, MAC TR-143 (January 1975) Project MAC, Mass Inst Tech.

[Tversky1974]

Tversky A, Kahneman D. Judgment under uncertainty: heuristics and biases, *Science*, 185 (18 September 1974) pp 1129-1131.

[van Melle1974]

van Melle W. Would you like advice on another horn, MYCIN project internal working paper, December 1974, Stanford University, Stanford California

[Waldinger1974]

Waldinger R, Levitt K N. Reasoning about programs, *Artificial Intelligence*, 5 (Fall 1974) pp 235-316.

[Warner1964]

Warner H R, Toronto A F, Veasy L G. Experience with Bayes' theorem for computer diagnosis of congenital heart disease, *Annals New York Academy of Science*, 115 (1964) pp 558-567.

[Warner1972]

Warner H R, Olmstead C M, Rutherford B D. HELP - a program for medical decision-making, *Computers and Biomedical Research*, 5 (1972) pp 65-74.

[Waterman1970]

Waterman D A. Generalization learning techniques for automating the learning of heuristics *Artificial Intelligence*, 1 (1970) pp 121-170.

[Winograd1972]

Winograd T. Understanding natural language, *Cognitive Psychology*, 3 (1972).

[Winograd1975]

Winograd T. Frame representations and the procedural/declarative controversy, in *Representation and Understanding*, Bobrow and Collins (eds.), Academic Press, 1975

[Woods1972]

Woods W A, Kaplan R M, Nash-Webber B. The lunar sciences natural language information system: final report, BBN Report 2378, June 1972, Bolt, Beranek and Newman, Cambridge, Mass.

[Zobrist1973]

Zobrist A L, Carlson F R. An advice-taking chess computer, *Scientific American*, 228 (June 1973) pp 92-105.